REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	11/15/2008	Final 8/1/2007 - 8/31/2008

4. TITLE AND SUBTITLE
Homogeneous Enclave Software Vs Controlled Heterogeneous Enclave Software

5. FUNDING NUMBERS
FA9550-07-1-0569

6. AUTHOR(S)
Schneider, Fred

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Cornell University
Office of Sponsored Programs
373 Pine Tree Road
Ithaca, NY 14850

8. PERFORMING ORGANIZATION REPORT NUMBER
55102

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Office of Scientific Research
875 N Randolph St.
Suite 325, Room 3112
Arlington VA 22203

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-SR-AR-TR-08-0512

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public Release; distribution is Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

A workshop of experts was convened on October 30, 2007 to consider the trade-offs associated with platform homogeneity in complex distributed systems. There were 18 speakers from industry and academia, and an additional 7 observers from AFCIO and AFOSR. The conclusion was that deploying a monocultures would be an effective way to defend against configuration attacks, that artificial diversity could then defend against some technology attacks (which monocultures amplify), and that some means would be required to defend against trust attacks. Research that needs to be pursued was identified.

14. SUBJECT TERMS
Monoculture, cybersecurity, automated diversity

15. NUMBER OF PAGES
20

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	unlimited

Workshop Report:

Security Risks from a Software Monoculture

Ken Birman and Fred B. Schneider

Department of Computer Science
Cornell University
Ithaca, New York 14853
U.S.A.

These slides contain findings and recommendations derived from a workshop conducted at IDA/Virginia on Oct 30, 2007 at the request of Chandersekaran Coimbatore for the office of the CIO, USAF.

USAF is imposing restrictions on procurements of desk top workstations. All new systems must run virtually the same exact versions of an operating system and suite of applications. The creation of such a monoculture has led to concern among security experts, because all systems in a monoculture are vulnerable to the same attacks. A single attacker could then inflict considerable damage with a relatively smaller investments than if, for example, the deployed systems were diverse.

A more careful look, however, reveals a far more complicated situation. People making decisions about procurement policy need to understand this more nuanced view in order to understand the implications of various procurement policy alternatives.

Recommendations Summary 1

Near-term operational:

- Proceed with planned transition to desktop monoculture.
  - This defends against configuration attacks, which are believed to be common and low-hanging fruit.
- Move quickly away from Windows XP to VISTA.
  - Windows VISTA (but not XP) supports address space randomization for the operating system. It creates a form of artificial diversity and thus better defends against technology attacks.
- Work with application providers so that they soon provide programs with artificial diversity.
  - In some cases, Windows VISTA address space randomization will work. In other cases, different means of artificial diversity, currently found in the research community, will have to be transitioned. Failure to make applications diverse risks vulnerability to technology attacks.

Recommendations Summary 2

Medium-term operational:

- Revisit and re-architect current enclave- structured networks to reduce vulnerability to trust attacks.
  - Deploying diverse systems increases the risk that a single compromised system can lead to an entire compromised enclave.
- Be conservative about deploying web standards. Develop defenses or ways to blunt the power of down-loaded scripting capabilities so common the web-based applications.
  - Standards constitute a monoculture and, by being widespread, will be scrutinized by attackers for vulnerabilities.
- Be conservative about server consolidation using virtual machines or other sub-OS multiplexing / isolation software.
  - Server consolidation code which might itself contain vulnerabilities. Moreover, compromising a VMM allows an attacker to compromise all hosted servers, so it is a high-payoff compromise.

Recommendations Summary 3

Long-term: Research needed:

- Develop new ways to automatically introduce diversity into software and into debugging and sharing of such diverse software.
  - Without such methods, the risk of technology attacks will remain high.
- Develop the means to measure diversity so there is a basis for choosing between architectures that involve monoculture and those with (inherently and artificially created) diverse components.
  - There is currently no principled way to reason about the architectural trade-offs that USAF is contemplating regarding monocultures, server consolidation, and thin clients.
- Overcome limitations of diversity---the defense is only probabilistic and only transforms attacks into availability outages.
  - Availability is crucial to combat and command/control.

What Constitutes a "Monoculture"?

monoculture: An environment in which the predominance of systems run apparently identical software components for some or all services.

- Such systems share vulnerabilities, hence they are at risk to rapid spread of a virus or other malware vector.

The term *monoculture* originates in the biological sciences, where it refers to a population entirely made up of a single organism (e.g., crop or animal). If monocultures today are rare in nature, it's because they can are so easily exterminated by a single pathogen. In contrast, a pathogen might destroy some members of a species that exhibits diversity but not all members—diversity thus helps ensure survival of a species.

Superficially, the USAF decision to select a single vendor platform (Microsoft Vista, Web-based email, etc) creates a monoculture. With all system instances uniform, the entire system is at risk to a rapidly-spreading virus or other attack vector. One should be especially concerned since that "entire system" might be the GIG or some other critical infrastructure for our military.

Study Goals

Assembled a group of experts who would:

- Identify security risks associated with widespread deployment of desktop workstations running a single (standard) software configuration: OS, mail, web browser, office tools, etc.
  - Ignored other benefits of monoculture: administration, education, maintenance and patch-management, skills transferability.
- Survey work in automated means for introducing heterogeneity in software systems:
  - What can be deployed today? In +3 yrs?
  - Identify impediments and research challenges.
- Discuss current and expected Internet-borne attacks and how they relate to:
  - software monocultures -versus-
  - heterogeneous systems.

A group of experts that could shed light on the issue was assembled.

The discussions were restricted to the security aspects of deploying a monoculture. There are, of course, many other reasons to favor a monoculture. When all systems are alike, then

•they are easier to manage (hence a smaller expense is incurred in system administration),

•less education is needed when a person moves from one part of the organization to another, and

•investments in education about how to use or manage the system can be amortized over a larger user base.

But these dimensions were not discussed at any length during the meeting. Our experts were more focused on security issues raised by the AF plans.

In addition to considering the immediate security consequences of deploying a monoculture, workshop participants were invited to reflect on research problems whose solution could materially reduce exposure to attackers and to discuss impediments to transitioning into the field known research results that could mitigate risks associated with monocultures.

Finally, we heard from experts on how malware propagates in the Internet and on experiences from those running large-scale data centers (which often do constitute a monoculture).

Study Non-Goals

This study does not treat some very important aspects of planned USAF changes to its computing infrastructure:

- Servers.
  - Server consolidation (E.g. use of VMM, physical aggregation of servers).
  - Client interaction with servers (E.g, thin client vs richer client interfaces to services).
- Network.
  - Means to limit or eliminate malware at Internet gateways.
  - Means to filter traffic inside a network to attenuate malware propagation.

USAF is currently discussing a rather bold restructuring of their computing infrastructure. This report addresses only one piece of the picture. There are good reasons to look at the other pieces in depth, too.

Server consolidation, for example, involves some subtle interesting trade-offs. Servers can be subject to closer scrutiny and their communications more closely monitored if they are located together. Yet, server consolidation typically involves using some sort of software-isolation layer (such as a virtual machine manager) to multiplex hardware among the different servers. This additional code adds a monoculture; by compromising a virtual machine manager, an attacker can get control of a collection of servers.

The virtues of thin client architectures are similarly nuanced. If clients store less data and have less access to servers, then damage from a compromised client can be contained. But thin clients connote communication which could enable the spread of malware to servers.

It is perhaps interesting to note that because the world has never see a thin client deployment with hundreds of thousands or millions of clients sharing a small number of data centers, this sort of consolidation is very much a research undertaking: an instance of the Air Force boldly going where nobody has ever gone before. This is worth mentioning because it implies that little is known about the risks of such deployments. Substantial research will be needed if the AF wishes to anticipate problems and proactively avoid them, rather than discovering them on the fly by doing experiments upon itself (with the help of adversaries, of course).

Such topics go well beyond the scope of our study but, frankly, no study could address them today. No panel of experts could possibly frame and address all the issues they raise. There are no domain experts with the necessary familiarity because such steps move outside of the commercial experience base creating a new

Topics Covered at Workshop

- Early monoculture investigations:
  - Stephanie Forrest (Univ of New Mexico)
  - Ravi Iyer (Univ of Ill)
- NSA perspective on monoculture risks:
  - Bill Unkenholz (NSA)
- Automated means for creating diversity:
  - Angelos Keromytis (Columbia Univ)
  - R. Sekar (Stony Brook Univ)
  - Ben Zorn (Microsoft Research)
  - Jack Davidson (Univ of Virginia)
  - John Knight (Univ of Virginia)

[continued...]

Early computer science investigations into monocultures were done by researchers investigating defenses that imitate a biological immune system and by researchers building fault-tolerant hardware. We heard about some of this work.

Mindful that NSA had deep insights into threats against DoD systems and into the vulnerabilities that monocultures bring, we made sure to get their perspective. This briefing, however, was done at an unclassified level.

A good deal of recent work in the security community has looked at methods to defend against various kinds of attacks by making random, but semantics-preserving, changes to a program before it executes. Such schemes are effective because attacks are often quite sensitive to a program's low-level implementation details—change these (while preserving the program's semantics) and the attack is defeated. We therefore heard from people with experience in designing and deploying these defenses.

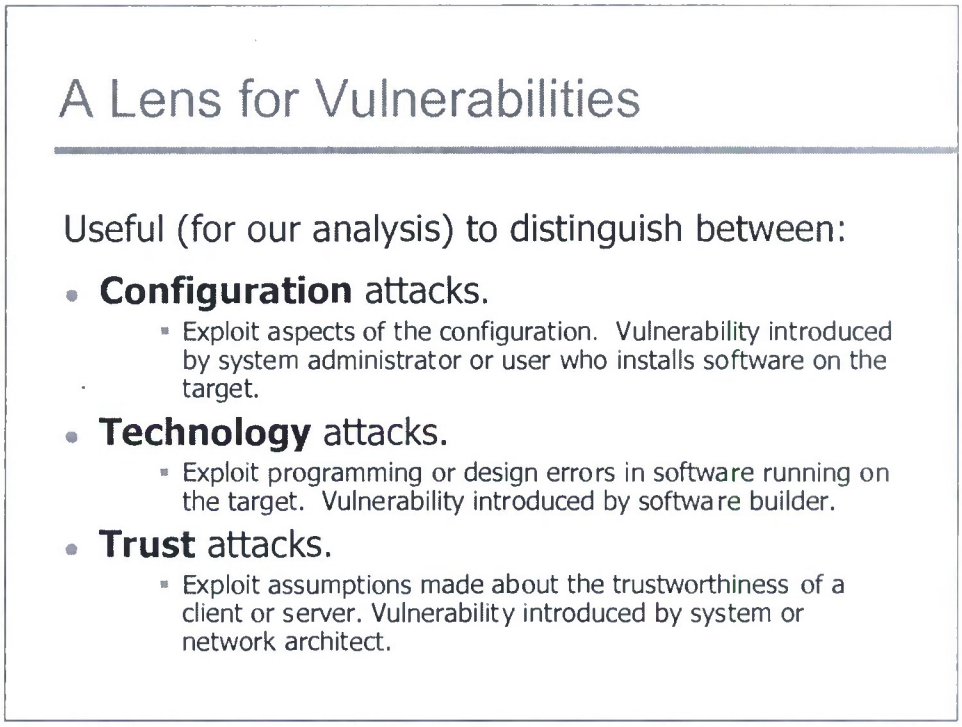
Workshop Coverage (con't)

- Attacks on automatic heterogeneity:
  - Hovav Shacham (U.C. San Diego)
- Network-borne malware:
  - Stefan Savage (U.C. San Diego)
  - Vern Paxon (U.C. Berkeley)
- Experience with real systems:
  - Tushar Chandra (Google)
  - Bruce Lindsay (IBM)
  - Armando Fox (U.C. Berkeley)
  - John Manferdelli (Microsoft)

Automatically created heterogeneity is not a panacea, though. Various schemes for introducing diversity have been successfully attacked. We learned about the specifics of one family.

Because a network is the likely vector for spreading an attack in a monoculture, we were briefed by people who have considerable experience in how such attacks spread through a network and how their spread can be attenuated by the systems comprising a network. The discussions covered not just with attacks against the operating system, but also attacks against middleware (such as databases), email systems, and end-user applications.

Finally, we heard from researchers who have some understanding about how large-scale server farms are managed and run. The last speaker offered the perspective of a software producer (Microsoft) whose products constitute a significant-sized monoculture but, though the use of automated diversity (address space randomization), is trying to reduce the vulnerability of that monoculture.



The consequences of deploying a monoculture are easier to understand if the problem is viewed in the right way. We submit that three significant classes of attacks should be treated independently: *configuration attacks*, *technology attacks*, and *trust attacks*. In the sequel, we discuss how each of these classes is affected by the presence of a monoculture.

Finding

Knowledge of threats must drive.

The unclassified community lacks a good understanding of the relative distribution of attack classes today (configuration vs technology vs trust) and of what attacks can be expected tomorrow. Researchers, in ignorance, might well be working on the wrong problems.

- Researchers work on things they hear about. They need to hear about the problems (*viz* classes of attack) of interest.
- A nation-state adversary is unlikely to launch subtle attacks (e.g., technology and trust) if obvious attacks (e.g., configuration) work. So today's attacks are not a good predictor for tomorrow's.

With only finite resources, one is forced to focus on only those threats ("motivated capable adversaries") perceived to be real. Knowledge of threats (such as available resources, expertise, likely targets, hoped for consequences) helps in predieting the kinds of attacks that are more likely.

Such knowledge about threats is typically not available to the majority of researchers, who work outside the classified setting. So most researchers investigate attacks that they believe are plausible and, absent authoritative information, they pursue defenses that involve tackling scientifically interesting research problems. System managers, absent authoritative information about threats, attempt to deploy defenses for attacks they believe are plausible, but often they are forced to deploy defenses imply because they are available.

We conclude that disseminating knowledge about threats is crucial to getting research done that is useful and to making intelligent choices about what defenses actually should get built and get deployed. The universe of possible attacks is huge and certainly larger than being observed in today's viruses, malware, and other exploits.

Finding

Monoculture: A sensible defense today.

A carefully constructed, fixed, system configuration would be an effective defense against configuration attacks.

- System configuration (today) is hard to get right and thus is best done by experts. Having one or a small number of "approved" configurations would allow that.
- Configuration attacks are considered "low hanging fruit" and thus likely are the dominant form of attack today.
  - NSA could validate this prediction.
- Configurations change not only because a system administrator installs software but also from a user visiting web sites or interacting with web services that cause software downloads.
  - To rule-out such downloads could be a serious limitation on system functionality. Such downloads often bring vulnerabilities, though.

Configuration errors are believed to be an overwhelming source of vulnerability in today's systems. Deploying a monoculture would help here, because a single locked-down, well understood configuration will have fewer vulnerabilities (by virtue of the care that can be invested in constructing that configuration). In a system embracing a high degree of diversity, every machine is a world unto itself and the chances are quite small that every machine will be correctly configured. So the chances that some machine could and would be compromised are quite high.

If you believe that configuration errors are a significant vulnerability today, then switching to a monoculture is the right thing to do. Data about whether configuration errors are, in fact, today a significant vulnerability was not made available to the workshop participants (but likely is known by NSA).

Finding

Monoculture: A risk for tomorrow.

- If configuration vulnerabilities are eliminated then attackers will turn to technology and trust vulnerabilities.
- A deployed monoculture is particularly vulnerable to technology attacks. *Need to deploy some defense!*
  - Defense: Use automated means to increase heterogeneity across the deployed software base.
    - Transforms some **integrity** and **confidentiality** violations into **availability** violations.
      - Attack typically ceases exec after 2-4 instruction [Keromytis]
      - But post-attack dump might convey obfuscation secret to attacker.
    - Security guarantees with this defense are only probabilistic.
    - Need high-quality confidential entropy for this defense.
    - Interpreted languages not handled.

If we eliminate opportunities for configuration attacks, then attackers will be forced to pursue other exploits. Any significant-sized software system is likely to have many (design and programming) flaws, and these often are the basis for technology attacks. Having deployed a monoculture, we must plan to focus attention on defending against attacks directed at the platform itself (Windows, is the case we focused on) or the applications running on it (Oracle, Webmail or Outlook, etc). In short, the attackers will focus on technology attacks, so --- as defenders --- we must too.

One defense here is the use automated tools to automatically introduce diversity into systems. However, the usual approach --- semantics preserving random code transformations --- simply transform attacks to compromise integrity or confidentiality into program crashes (with some probability). Systems where availability is crucial are not helped, nor are systems where a defense that works with certainty is needed.

Indeed, a tool for automatic creation of diversity are even part of Windows Vista, as shipped today. But many of the techniques are not yet available in a form that the AF can deploy today, and in some cases additional research is required. Investing in that work today will allow the techniques to be developed and deployed in time.

Finding

Create diversity automatically

Heterogeneity can be introduced at compile or load time by making "random" changes to:

- Address space and storage layout:
  - Allocation of variables on stack
  - Allocation of variables in heap

  Done today in Microsoft VISTA OS but not for Windows applications.
- Instruction set opcodes
  - all instructions vs system calls
  - limited use when attacker is on the same machine.
- System state
  - encoding of values

Heterogeneity also arises naturally from non-determinacy in parallel systems.

Various semantics-preserving random program transformations have been investigated by researchers. These involve pre-processing a program (either in source or binary form) before execution.

In addition, non-determinacy in the order that events occur in a large asynchronous systems leads to diversity is the system state (which, in turn, can help defend in a limited way against certain attacks).

Recommendation:

Air Force must **today** invest to:

- Understand how to automatically introduce runtime diversity.
- Take steps to ensure these solutions can and will quickly transition to practice. Potential issues include:
  - Code sharing and debugging
  - Coping with availability outages intrinsic in diversity defenses.
  - Lack of diversity metrics that have predictive value regarding classes of attacks (when defined concretely or abstractly).
- Plan for defense in depth:
  - Explore and use network-monitoring techniques to stop virus outbreaks from starting or spreading.
  - Be able to monitor system-wide health picture to detect (and control) malware outbreaks and propagation without disrupting key AF services.

Absent such steps, there is a serious risk associated with current efforts to deploy a monoculture.

The AF is urged to appreciate the urgency of taking proactive measures that will mitigate the risks of homogeneity. The very act of deploying a monoculture will cause attackers to change the focus of their attention away from configuration attacks and toward technology attacks.

Automatically creating diversity is a promising defense, but various barriers exist today for its widespread deployment. Schemes for code-sharing and debugging must be modified. There are no good solutions for dealing with availability outages that the diversity defense can cause. And there is no science base for comparing and analyzing various diversity defenses.

Technology attacks against a monoculture also can be blocked by slowing the spread of the attack. Such defenses include various system-wide (and automatic, because attacks can spread quite rapidly) monitoring and control schemes.

If the AF is proactive, the risks of monoculture deployment can be contained, while the benefits could result in an overall increase in security. Yet we worry that the AF itself is sufficiently aware of the risks of doing nothing.

Especially acute is the need for further research on some of the most promising technology paths --- paths that have been identified by early research in the area, but for which more work will be necessary in order to better understand the options, the details of how they might be deployed and used on a large scale, and the best ways of implementing them given the "black box" nature of much of the software that will run on AF platforms.

No unclassified research program on these topics currently exists within DoD (including AF, DARPA, etc) or DHS, and the absence of such research is a recipe for problems down the road (ie. 5+ years). The need to impact commercial standards and the widespread use of COTS technology means that a purely classified solution cannot be adequate, even if classified work is underway.

Finding

Diversity: Also a risk for tomorrow.

Principals (processes, programs, hosts) within an enclave will typically trust other principals in that enclave.

- A compromised principal P within an enclave constitutes a launch point for attacking others (i.e., those that trust P).
- With diversity, the attacker has many choices of what system to compromise, since all have different vulnerabilities.
- An attack that succeeds against any system in the enclave is then easily leveraged to compromise the rest of the enclave.

Such **trust attacks** are increasingly seen today: E.g., cross-site scripting.

- Defense: Re-architect enclave---use of "least privilege" authorization.
- Defense: Use of "trusted computing" to provide accountability for trajectory of service invocations and prevent request hijacking.

The third class of attacks --- trust attacks --- exhibits an insidious interaction with diversity.

One way to organize a networked system is in terms of enclaves, where machines inside an enclave trust each other more than they trust machines outside that enclave. Thus, if any machine inside an enclave is compromised then it is relatively easy for that machine to serve as a launching pad for successfully compromising other machines in the enclave.

When an enclave comprises a diverse set of machines, then the attacker's job is actually made easier. Different machines have different vulnerabilities, and compromising any single machine provides an entry point from which (misplaced) trust can be exploited to subvert other machines in the enclave. So here the diversity actually works to an attackers advantage.

Attacks involving misplaced trust are increasingly common. Cross-site scripting attacks depend on misappropriating trust, and these attacks are easily hidden in systems built as web services.

Recommendation

Study USAF enclave architectures and develop approaches to reduce risk that attacks can spread from one compromised element to another:

- Re-architect enclaves to limit intra-enclave trust as a basis for authorization.
- Develop schemes to reduce risk that attacks can spread from one compromised element in an enclave to another.

The practice of structuring networked systems in terms of enclaves needs to be revisited once attackers no longer have access to the low-hanging fruit of configuration attacks. And the obvious defense against technology attacks, which is diversity, would seem to make enclave architectures even more easily attacked.

Finding

Standards are a form of monoculture.

Webmail and other browser solutions implement standards:

- Javascript
- AJAX

Successful standards (by definition) become widespread; attackers thus have incentives to invest in exploits based on vulnerabilities in a successful standard.

Successful standards bring the exposures of deploying a monoculture.

Any standard, by definition, creates a kind of monoculture associated with the ubiquitous deployment of interfaces and services implementing that standard. This is particularly true for Web Services, which deploy AJAX and JavaScript as executable software standards and XML for representing data. Here, technology attacks not only could involve exploiting the semantics of system internals but also might involve the interfaces themselves. The diversity defense is not (currently) an option for defending against attacks that exploit the (misguided) semantics of an interface.

For example, the future is likely to bring attacks that exploit web-based email. This can be predicted because the emerging AJAX standard, which is part of the GIG platform standards associated with Web Services, is powerful enough to create a full-scale network computing platform. In effect, email becomes a kind of operating system that can run applications, download code from various places on the network, and access files and other local resources. The reasons for including these AJAX features seemed obvious to the people developing them: they allow email to support attachments, to render rich content, etc. Yet the features create new kinds of vulnerabilities that are poorly understood today.

This email example is just one of many, but it highlights a serious concern: What seems like a sensible operational decision --- supporting email from anywhere and to anywhere and supporting web services more generally --- creates potential new vulnerabilities.

It would be foolish for the AF turn away from email, or from the GIG standards. The AF is wise to adopt standards, but must proactively work to anticipate vulnerabilities these create and act to mitigate them. Because these standards cross platforms and applications, the longe- term threat could even be greater than that associated with widespread use of Windows as an operating system platform on client machines.

Recommendation

- Air Force should move cautiously in deploying software systems based on standards.
- Standards that support downloaded code are particularly worrisome, since downloaded code can be a vector for attacks.
  - Technology needed to better isolate downloaded code, yet allow it to interact with its environment.
  - Research needed into ways of securing applications constructed by composing web services (a successful standard).

Alternative of "outlawing" these technologies not feasible.

It would be foolish for the AF turn away from email, or from the GIG standards. The AF is wise to adopt standards, but must proactively work to anticipate vulnerabilities these create and act to mitigate them. Because these standards cross platforms and applications, the longer- term threat could even be greater than that associated with widespread use of Windows as an operating system platform on client machines.

Summary

- Deploying a monoculture solves some problems but creates others.
- Monocultures can arise by design (e.g., procurement) or by accident (adoption of successful standards). The Air Force is pursing a trajectory that encompasses both.
- Research does exist to improve the cyber-defense posture against certain risks monocultures create.
  - Air Force investments are required before that work can be transitioned to practice.
- Research is needed to address other problems that will arise.
  - We can predict what kinds of vulnerabilities attackers will target as easy vulnerabilities are closed. Air Force must be positioned to deploy defenses for these when the need arises.

The monoculture question turns out to be quite subtle.

Deploying a monoculture will help defend against configuration attacks. But that success is likely to propel attackers toward launching other attacks---technology attacks and trust attacks. Defenses for these will need to be fielded.

Researchers have developed some defenses against certain technology attacks. These defenses are based on automatically introducing diversity and on various network-filtering schemes that attenuate the spread of an attack. Some, but not all, of this work has been transitioned to practice. So investments will be needed to transition other solutions and for research to address limitations that are discovered and to construct a science base for understanding and evaluating the alternatives.

Bolt-on defenses are not available for trust attacks. Research is badly needed here.

Bibliography

Barrantes, Ackley, Forrest, Palmer, Stefanovic, and Zovi. Randomized Instruction Set Emulation to Disrupt Binary Code Injection Attacks. *Proceedings of the 10th Annual ACM Conference on Computer and Communications Security* (CCS '03), ACM Press, October 2003, 281-289.

Barrantes, Ackley, Forrest, and Stefanovic. Randomized instruction set emulation. *ACM Transactions on Information System Security* 8, 1 (2005), 3-40.

Emery D. Berger and Benjamin G. Zorn. DieHard: Probabilistic Memory Safety for Unsafe Languages. *Department of Computer Science Technical Report 05-65*, University of Massachusetts Amherst, 2005.

S. Bhatkar, D.C. DuVarney, and R. Sekar. Address obfuscation: An efficient approach to combat a broad range of memory error exploits. *Proceedings of the 12th Annual USENIX Security Symposium*, 2003, 105-120.